Brief Announcement: New Bounds for the Controller Problem

Yuval Emek[*]
School of Electrical Engineering
Tel Aviv University
Tel Aviv, Israel
yuvale@eng.tau.ac.il

Amos Korman[†]
CNRS and Université Paris Diderot
Paris 7, France
amos.korman@gmail.com

ABSTRACT

The (M,W)-*controller*, originally studied by Afek, Awerbuch, Plotkin, and Saks, is a basic distributed tool that provides an abstraction for managing the consumption of a global resource in a distributed dynamic network. We establish new bounds on the message complexity of this tool based on a surprising connection between the controller problem and the *monotonic labeling problem.*

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems—distributed applications

General Terms: Algorithms

Keywords: controller, dynamic networks

1. INTRODUCTION

Considered as one of the elementary and fundamental tools in distributed computing (cf. [2]), *controllers* (originally studied in [1] and later in [10]) provide an abstraction for global resource consumption management. In particular, controllers serve as a key ingredient in the state of the art solutions for various problems such as majority commitment in a network where some of the nodes failed before the algorithm started, routing (and other informative labeling problems) in dynamic trees, and dynamic name assignment.

The (M,W)-*controller* problem. We consider a distributed network operating in an asynchronous environment. Initially, a set of *permits* resides at some designated node called the *root*. A subset of permits may be delivered from node u to any of its neighbors v by sending a single message from u to v (this message essentially encodes the number of permits that are being delivered). Therefore throughout the execution the permits are distributed among the nodes of the network and different nodes may hold different numbers of permits. The input to the controller arrives online in the form of *requests* presented at arbitrary nodes. When a request is presented at node u, the controller must respond within finite time in one of the following two manners: (1) it may *grant* the request by delivering a permit to u in which case the permit is eliminated from the network (corresponding to consuming one unit of the global resource at node u); or (2) it may *reject* the request.

In an (M,W)-controller, the number of permits that initially reside at the root is M, indicating that at most M requests can be granted. On the other hand, the (M,W)-controller may reject a request only if it is certain that at most W permits eventually remain in the network. In other words, if an (M,W)-controller rejects a request, then it is guaranteed that at least $M-W$ requests were already granted (or will be granted within finite time).

It is assumed that a spanning tree T rooted at some node r is maintained in the network and that the controller relies on the links of T for communication. The global resource whose consumption is managed by the controller may be of various types. However, since the concept of an (M,W)-controller finds many applications in dynamic networks, a special attention has been given to the case where a request presented at node u represents the desire to perform a topology change at the vicinity of u. Such a request is referred to as a *topological request*. Specifically, the topology changes considered in this context are: (i) inserting a new child of u as a leaf in T; (ii) inserting a new child of u as an internal node in T by subdividing a link that connects u to one of its children; and (iii) deleting a child v of u and turning the children of v into children of u (the root r is never deleted). In all three cases the actual topology change is assumed to occur once the topological request is granted a permit.

The number of nodes that ever existed in the network (including the deleted ones) is denoted by N. Note that N cannot exceed the initial network size by more than M since the insertion of every new node should be granted a permit by the controller (in fact, the combined number of node insertions and deletions is at most M).

The efficiency of an (M,W)-controller is measured by means of its *message complexity*, namely, the total number of messages sent during the execution. This is usually expressed as a function of M, W, and N. Afek et al. [1] construct the first (M,W)-controller which admits message complexity $O(N \log^2 N \log \frac{M}{W+1})$. Their controller only supports the insertion of leaves. Korman and Kutten [10] introduce an (M,W)-controller with a similar message complexity which supports all three types of topology changes (i.e., the insertion of leaves, the insertion of internal nodes, and the deletion of nodes). Both the (M,W)-controller of [1] and that of [10] are implemented by first constructing an $(M,M/2)$-controller with message complexity $O(N \log^2 N)$,

[*]Partially supported by the Israel Science Foundation, grant 664/05.

[†]Supported in part by the ANR project ALADDIN, by the INRIA project GANG, and by COST Action 295 DYNAMO.

and then invoking it in $O(\log \frac{M}{W+1})$ iterations. The controller of [10] encodes each message using $O(\log N)$ bits, while the (more restricted) controller of [1] encodes each message using $O(\log \log N)$ bits.

On the negative side, it is easy to see that an $\Omega(N)$ term in the message complexity of any (M, W)-controller is inevitable. (In the case of an N-node path, for example, merely delivering a permit from the root to a request presented at the other end requires N messages.) However, no non-trivial lower bounds were previously known.

The monotonic labeling problem. Vital to our techniques is the *monotonic labeling problem.* In this (centralized) problem n distinct elements from some dense totally ordered set S (e.g., the real numbers) are introduced, one at a time. Upon introduction, each element $x \in S$ should be assigned with a *label* $\lambda(x)$ taken from some discrete totally ordered set L of adequate ($|L| \geq n$), yet limited, cardinality (e.g., the integers $1, \ldots, |L|$). The order of the labels must agree with the order of the elements, that is, for every two elements $x, y \in S$, if $x < y$, then $\lambda(x) < \lambda(y)$. Therefore from time to time some previously introduced elements must be *relabeled* to "make room" for new elements. The objective of a monotonic labeling algorithm is to minimize the total number of labeling operations (including relabeling previously introduced elements). This is typically measured as a function of n and with respect to the cardinality of the label set L (clearly, the problem becomes easier as $|L|$ grows).

The monotonic labeling problem is essentially introduced in [9] and studied further in [5, 12, 11, 7, 3, 8, 4, 6], mainly in the context of maintaining an ordered data structure. With label sets of cardinality n, $n(1+\epsilon)$, and $n^{1+\epsilon}$, where ϵ is any positive constant, the known upper bounds for the number of labeling operations are $O(n \log^3 n)$ [3], $O(n \log^2 n)$ [9, 12, 4], and $O(n \log n)$ [5, 11, 7]. An $\Omega(n \log n)$ lower bound for the number of labeling operations with label sets of cardinality polynomial in n is established in [6], thus showing that the upper bound of [5, 11, 7] is tight. Based on a lower bound established in [8] for the special class of *smooth* algorithms, the authors of [8, 6] conjecture that any monotonic labeling algorithm with $O(n)$ labels requires $\Omega(n \log^2 n)$ labeling operations, hence the upper bound of [9, 12, 4] is also tight.

2. PROGRESS

In this work we establish new bounds on the message complexity of the controller problem. As a warm up, we first prove a simple lower bound stating that any (M, W)-controller must send $\Omega(N \log \frac{M}{W+1})$ messages. Although this lower bound is meaningful for small values of W, it is not very informative when W is proportional to M, which is the typical case in many applications of the controller problem.

Subsequently, we turn our attention to the case where W is proportional to M and prove that for every constant $\epsilon > 0$, an $(M, M(1-\epsilon))$-controller on a dynamically growing path of initial size M must admit message complexity $\Omega(M \log M) = \Omega(N \log N)$. This lower bound is obtained due to a surprising reduction from the (centralized) monotonic labeling problem to the (distributed) controller problem. Through this reduction, the $\Omega(n \log n)$ lower bound on the number of labeling operations that must be performed by any monotonic labeling algorithm with a label set of cardinality polynomial in n translates to the desired $\Omega(N \log N)$ lower bound on the message complexity of a controller. In fact, the reduction holds for monotonic labeling algorithms with label sets of cardinality $O(n)$, and therefore as it turns out, under the conjecture of [8, 6], we obtain a tight $\Omega(N \log^2 N)$ lower bound on the message complexity of any $(M, M(1-\epsilon))$-controller.

Both our lower bounds hold even when the message size is unbounded. Furthermore, they do not rely on concurrency considerations, and therefore remain valid even if the system is synchronous and the requests are "spaced in time" so that the next request is presented only after the controller finished handling all previous ones.

As previously mentioned, the proof of the $\Omega(N \log N)$ lower bound (and also of the conjectured tight $\Omega(N \log^2 N)$ lower bound) relies on a network of initial size M which, in particular, implies that $N = \Theta(M)$. It turns out that this is no coincidence: such a lower bound cannot hold if M is much smaller than N. We prove it by constructing a novel $(M, M/2)$-controller with message complexity $O(N \log^2 M)$. Apart from demonstrating the inherent limitation of our lower bound proof technique, the new controller is interesting as it can be generalized (c.f. Section 5 in [1]) to an (M, W)-controller with message complexity $O(N \log^2 M \log \frac{M}{W+1})$, thus exhibiting an asymptotic improvement to the state of the art in the case that M is sub-polynomial in N. Moreover, the structure of our new controller is completely different than the previously known controllers and bears an independent algorithmic interest.

3. REFERENCES

[1] Y. Afek, B. Awerbuch, S.A. Plotkin and M. Saks. Local management of a global resource in a communication network. *J. ACM*, 43:1–19, 1996.

[2] Y. Afek and M. Ricklin. Sparser: a paradigm for running distributed algorithms. *J. Algorithms*, 14(2):316–328, 1993.

[3] A. Andersson and T. W. Lai. Fast updating of well-balanced trees. In *SWAT*, pages 111–121, 1990.

[4] M.A. Bender, R. Cole, E.D. Demaine, M. Farach-Colton and J. Zito. Two simplified algorithms for maintaining order in a list. In *ESA*, pages 152–164, 2002.

[5] P. F. Dietz. Maintaining Order in a Linked List. In *STOC*, pages 122–127, 1982.

[6] P. F. Dietz, J. I. Seiferas, and J. Zhang. A tight lower bound for online monotonic list labeling. *SIAM J. Discrete Math.* 18(3):626–637, 2004.

[7] P. F. Dietz and D. D. Sleator. Two algorithms for maintaining order in a list. In *STOC*, pages 365–372, 1987.

[8] P. F. Dietz and J. Zhang. Lower bounds for monotonic list labeling. In *SWAT*, pages 173–180, 1990.

[9] A. Itai, A. Konheim, and M. Rodeh. A sparse table implementation of priority queues. In *ICALP*, pages 417–431, 1981.

[10] A. Korman and S. Kutten. Controller and estimator for dynamic networks. In *PODC*, pages 175–184, 2007.

[11] A. K. Tsakalidis. Maintaining order in a generalized linked list. *Acta Inform.*, 21:101–112, 1984.

[12] D. Willard. Maintaining dense sequential files in a dynamic environment. In *STOC*, pages 114–121, 1982.